# 最牛B的 Linux Shell 命令

英文原文： Top Ten One-Liners from CommandLineFu Explained

译者: boypt / isspy / riku / 制作：riku
发布日期: 2010 年 9 月10 日

*注*：在经过作者 @isspy 的同意下，此系列文章被转载于 Wow! Ubuntu（其中第三部分由 riku 编译）。个人认为作为 Linux 用户，熟练掌握 CLI 命令是一个比较基本的能力，而这篇连载文章提供了更深层次的内容，可以帮助大家学习到更多有用的内容。

转载地址: http://wowubuntu.com/tag/linuxshell

# 编者按

本文编译自 Catonmat 的系列文章 Top Ten One-Liners from CommandLineFu Explained 。作为一个由用户推荐最有用shell命令的网站，其记录了数以万计的各色shell命令，其中不乏相当实用和有趣的，本文就要细数当中获投票最高的一些命令，从其中取材并加以细释，希望读者能从中受益。

# 引言

Shell作为Unix系操作系统当中最有魅力且不可或缺的组件，经过数十载的洗礼不仅没有被淘汰，而且愈加变得成熟稳健，究其原因，大概因为它是个非常稳固的粘合剂，能够把大量功能强大的组件任意配搭，总能很好很快地完成用户的任务。

本文的一些命令很可能看起来是"雕虫小技"，我们只好仰慕一下Shell大牛了，但是有些细节我会稍加发掘加以说明，遇到有趣的地方希望能博您一笑了。

# === 第一部分 ===

**1.以SUDO运行上条命令**

```
$ sudo !!
```

大家应该都知sudo，不解释。但通常出现的情况是，敲完命令执行后报错才发现忘了sudo。这时候，新手用户就会：按上箭头，按左箭头，盯着光标回到开始处，输入sudo，回车；高手用户就蛋定多了，按Ctrl-p，按Ctrl-a，输入sudo，回车。

这里介绍这个是天外飞仙级别的，对，就直接sudo !!。

当然这几种解决方式效果是完全一样的，只是款不一样，嗯，不解释。

两个感叹号其实是bash的一个特性，称为事件引用符（event designators）。!!其实相当于!-1，引用前一条命令，当然也可以!-2，!-50。默认情况下bash会在~/.bash_history文件内记录用户执行的最近500条命令，history命令可以显示这些命令。

关于事件引用符的更多用法可以深入阅读 The Definitive Guide to Bash Command Line History。

**2.以HTTP方式共享当前文件夹的文件**

```
$ python -m SimpleHTTPServer
```

这命令启动了Python的SimpleHTTPServer模块，考虑到Python在绝大多数的Linux发行版当中都默认安装，所以这个命令很可能是最简单的跨平台传文件的方法。

命令执行后将在本机8000端口开放HTTP服务，在其他能访问本机的机器的浏览器打开ttp://ip:8000即打开一个目录列表，点击即可下载。

**3.在以普通用户打开的VIM当中保存一个ROOT用户文件**

```
:w !sudo tee %
```

这题目读起来纠结，其实是很常见的，常常忘记了sudo就直接用vim编辑/etc内的文件，（不过也不一定，vim发现保存的文件无法保存时候会提示）等编辑好了，保存时候才发现没权限。曲线方法是先保存个临时文件，退出后再sudo cp回去。不过实际上在vim里面可以直接完成这个过程的，命令就是如此。

查阅vim的文档（输入:help :w），会提到命令:w!{cmd}，让vim执行一个外部命令{cmd}，然后把当前缓冲区的内容从stdin传入。

tee是一个把stdin保存到文件的小工具。

而%，是vim当中一个只读寄存器的名字，总保存着当前编辑文件的文件路径。

所以执行这个命令，就相当于从vim外部修改了当前编辑的文件，好完工。

**4.切换回上一个目录**

```
$ cd -
```

应该不少人都知道这个，横杆-代表上一个目录的路径。

实际上cd -就是cd $OLDPWD的简写，bash的固定变量$OLDPWD总保存着之前一个目录的路径。

相对地，$PWD总保存着当前目录的路径。这些变量在编写shell脚本时候相当有用。

**5.替换上一条命令中的一个短语**

```
$ ^foo^bar^
```

又是另外一个事件引用符（event designator），可以把上一条命令当中的foo替换成bar。

在需要重复运行调试一道长长的命令，需要测试某个参数时候，用这个命令会比较实用；但多数人会首先选择按上箭头提出上道命令，再移动光标去修改某参数，这样更直观，但效率上就不够使用引用符高，而且在脚本中用这个方法可以简化很多。

这道命令的原始样式应该是这样的:

```
!!:s/foo/bar/
```

本文一开始介绍过!!，后面的一段大家应该很熟悉，vim、sed的替换操作都是这样的语法。

关于事件引用符的更多用法可以深入阅读 The Definitive Guide to Bash Command Line History

**6.快速备份一个文件**

```
$ cp filename{,.bak}
```

这道命令把filename文件拷贝成filename.bak，大家应该在一些比较复杂的安装教程里面见过这样的用法。其原理就在于bash对大括号的展开操作，filename{,.bak}这一段会被展开成filename filename.bak再传给cp，于是就有了备份的命令了。

大括号在bash里面是一个排列的意义，可以试试这个：

```
$ echo {a,b,c}{a,b,c}{a,b,c}
```

将输出三个集合的全排列:

```
aaa aab aac aba abb abc aca acb acc
baa bab bac bba bbb bbc bca bcb bcc
caa cab cac cba cbb cbc cca ccb ccc
```

关于shell当中的集合操作，可深入阅读"Set Operations in the Unix Shell"

**7.免密码SSH登录主机**

```
$ ssh-copy-id remote-machine
```

这个命令把当前用户的公钥串写入到远程主机的~/.ssh/authorized_keys内，这样下次使用ssh登录的时候，远程主机就直接根据这串密钥完成身份校验，不再询问密码了。前提是你当前用户有生成了公钥，默认是没有的，先执行ssh-keygen试试吧！

这个命令如果用手工完成，是这样的：

```
your-machine$ scp ~/.ssh/identity.pub remote-machine:
your-machine$ ssh remote-machine
remote-machine$ cat identity.pub >> ~/.ssh/authorized_keys
```

如果你想删掉远程主机上的密钥，直接打开authorized_keys，搜索你的用户名，删除那行，即可。

**8.抓取LINUX桌面的视频**

```
$ ffmpeg -f x11grab -s wxga -r 25 -i :0.0 -sameq /tmp/out.mpg
```

我们在一些视频网站上看到别人的3D桌面怎么怎么酷的视频，通常就是这么来的，ffmpeg可以直接解码X11的图形，并转换到相应输出格式。

ffmpeg的通常用法是，根据一堆参数，输出一个文件，输出文件通常放最后，下面解析下几个参数：

-f x11grab 指定输入类型。因为x11的缓冲区不是普通的视频文件可以侦测格式，必须指定后ffmpeg才知道如何获得输入。
-s wxga 设置抓取区域的大小。wxga是1366*768的标准说法，也可以换成-s 800×600的写法。
-r 25 设置帧率，即每秒抓取的画面数。
-i :0.0 设置输入源，本地X默认在0.0
-sameq 保持跟输入流一样的图像质量，以用来后期处理。

至于其他ffmpeg的用法，可以参考下面两篇文章：

How to Extract Audio Tracks from YouTube Videos
Converting YouTube Flash Videos to a Better Format with ffmpeg

# === 第二部分 ===

**1.用你最喜欢的编辑器来敲命令**

```
command <CTRL-x CTRL-e>
```

在已经敲完的命令后按 <CTRL-x CTRL-e> ，会打开一个你指定的编辑器（比如vim，通过环境变量$EDITOR指定），里面就是你刚输入的命令，然后爱怎么编辑就怎么编辑吧，特别是那些参数异常复杂的程序，比如mencoder/ffmpeg，一个命令动辄3、4行的，要修改其中的参数，这个方法最合适不过了，保存退出后自动执行这个程序。

实际上这是readline库的功能，在默认情况下，bash使用的是emacs模式的命令行操作方式， <CTRL-x CTRL-e>是调用这个功能的一个绑定。如果你习惯使用vi模式，按 <ESC v> 可以实现同样功能。

如果你喜欢别的编辑器，可以在~/.bashrc里面放上比如export EDITOR=nano的命令。另外一个修改命令的方法是使用fc命令（Fix Command），在编辑器里面打开上一句命令。我们的第一辑连载提过一个^foo^bar^命令可以用fc来实现：fc -s foo=bar。

**2.清空或创建一个文件**

```
> file.txt
```

>在shell里面是标准输出重定向符，即把（前部个命令的）命令行输出转往一个文件内，但这里没有"前部命令"，输出为空，于是就覆盖（或创建）成一个空文件了。

有些脚本的写法是:>file.txt，因为:是bash默认存在的空函数。

单纯创建文件也可以用$touch file.txt，touch本来是用作修改文件的时间戳，但如果文件不存在，就自动创建了。

**3.用SSH创建端口转发通道**

```
ssh -N -L2001:remotehost:80 user@somemachine
```

这个命令在本机打开了2001端口，对本机2001端口的请求通过somemachine作为跳板，转到remotehost的80端口上。

实现效果跟术语反向代理是相似的，实际上就是端口转发，注意上面的描述涉及了3台主机，但当然somemachine可以变成localhost。

这个命令比较抽象，但有时候是很有用的，比如因为众所周知的原因国内的IP的80端口无法使用，又或者公司的防火墙只给外网开了ssh端口，需要访问内部服务器

一个web应用，以及需要访问某些限定了来源IP的服务，就可以用上这个方法了。

举一个具体例子，运行：

```
ssh -f -N -L 0.0.0.0:443:twitter.com:443 shell.cjb.net
ssh -f -N -L 0.0.0.0:80:twitter.com:80 shell.cjb.net
```

然后在/etc/hosts里面添加127.0.0.1 twitter.com，好吧剩下的你懂的。

当然通常做这个功能的反向代理，应该要用squid、nginx之类，ssh就算是轻量级的尝试吧！

**4.重置终端**

```
reset
```

如果你试过不小心cat了某个二进制文件，很可能整个终端就傻掉了，可能不会换行，没法回显，大堆乱码之类的，这时候敲入reset回车，不管命令有没有显示，就能回复正常了。

实际上reset命令只是输出了一些特殊字符，我们看BusyBox里面最简单的reset程序的实现：

```
printf("\033c\033(K\033[J\033[0m\033[?25h");
```

输出的这些字符对Shell是有特殊意义的：

\033c: "ESC c" – 发送重置命令;
\033(K: "ESC ( K" – 重载终端的字符映射;
\033[J: "ESC [ J" – 清空终端内容;
\033[0m: "ESC [ 0 m" – 初始化字符显示属性;
\033[?25h: "ESC [ ? 25 h" – 让光标可见;

其中字符显示属性经常用来设定打印字符的颜色等，可参考这个博文。

**5.在午夜的时候执行某命令**

```
echo cmd | at midnight
```

说的就是at这个组件，通常跟cron相提并论，不过at主要用于定时一次性任务，而cron定时周期性任务。

at 的参数比较人性化，跟英语语法一样，可以tomorrow, next week之类的，详细的查看手册man at。

**6.远程传送麦克风语音**

dd if=/dev/dsp | ssh username@host dd of=/dev/dsp

没错就是实现一个喊话器的功能。

/dev/dsp是Linux下声卡的文件映射（Digital Signal Proccessor），从其中读数据就是录音，往里面写数据就是播放，相当简单！

dd是常用的数据拷贝程序，如果不同时指定if、of，就直接使用stdin/stdout来传输。

如果你没有远程主机，可以试试这样：

dd if=/dev/dsp of=/dev/dsp

直接回放麦克风的声音，只是有一点延时。

但是如果有别的程序正在使用声卡，这个方法就不凑效了，因为一般的声卡都不允许多个音频流同时处理，可以借用alsa组件的工具，arecord跟aplay:

arecord | ssh username@host aplay

本地回放就是：

arecord | aplay

如果你想吓吓别人：

cat /dev/urandom | ssh username@host aplay

**7.映射一个内存目录**

mount -t tmpfs -o size=1024m tmpfs /mnt/ram

这个命令开了一块1G内存来当目录用。不过放心，如果里面没文件，是不会占用内存的，用多少占多少。

不过一般来说没必要手动挂载，因为多数发行版都会在fstab内预留了一个内存目录，挂载在/dev/shm，直接使用即可；

最常见的用途是用内存空间来放Firefox的配置，可以让慢吞吞的FF快很多，参见Shellex的博文：用tmpfs让Firefox在内存中飞驰，以及后来的改进：tmpfs Firefox II，其中提到的脚本来自speeding up firefox with tmpfs and automatic rsync。那个破烂LinuxQQ也可以用这个方法，减少因为大量磁盘IO导致的问题。

**8.用DIFF对比远程文件跟本地文件**

ssh user@host cat /path/to/remotefile | diff /path/to/localfile -

diff通常的用法是从参数读入两个文件，而命令里面的-则是指从stdin读入了。

善用ssh可以让web开发减少很多繁琐，还有比如sshfs，可以从编辑-上传-编辑-上传的人工循环里面解脱出来。

**9.查看系统中占用端口的进程**

netstat -tulnp

Netstat是很常用的用来查看Linux网络系统的工具之一，这个参数可以背下来：

-t: 显示TCP链接信息
-u: 显示UDP链接信息
-l: 显示监听状态的端口
-n: 直接显示ip，不做名称转换
-p: 显示相应的进程PID以及名称（要root权限）

如果要查看关于sockets更详细占用信息等，可以使用lsof工具。

# === 第三部分 ===

**1. 更友好的显示当前挂载的文件系统**

```
mount | column -t
```

这条命令适用于任何文件系统，column 用于把输出结果进行列表格式化操作，这里最主要的目的是让大家熟悉一下 columnt 的用法。

下面是单单使用 mount 命令的结果：

```
$ mount
/dev/root on / type ext3 (rw)
/proc on /proc type proc (rw)
/dev/mapper/lvmraid-home on /home type ext3 (rw,noatime)
```

而加了 column -t 命令后就成为这样了：

```
$ mount | column -t
/dev/root on / type ext3 (rw)
/proc on /proc type proc (rw)
/dev/mapper/lvmraid-home on /home type ext3 (rw,noatime)
```

另外你可加上列名称来改善输出结果

```
$ (echo "DEVICE - PATH - TYPE FLAGS" && mount) | column -t

DEVICE - PATH - TYPE FLAGS
/dev/root on / type ext3 (rw)
/proc on /proc type proc (rw)
/dev/mapper/lvmraid-home on /home type ext3 (rw,noatime)
```

列2和列4并不是很友好，我们可以用 awk 来再处理一下

```
 $ (echo "DEVICE PATH TYPE FLAGS" && mount | awk '$2=$4="";1') | column -t

DEVICE PATH TYPE FLAGS
/dev/root / ext3 (rw)
/proc /proc proc (rw)
```

```
/dev/mapper/lvmraid-home /home ext3 (rw,noatime)
```

最后我们可以设置一个别名，为 nicemount

```
$ nicemount() { (echo "DEVICE PATH TYPE FLAGS" && mount |
awk '$2=$4="";1') | column -t; }
```

试一下

```
$ nicemount
DEVICE PATH TYPE FLAGS
/dev/root / ext3 (rw)
/proc /proc proc (rw)
/dev/mapper/lvmraid-home /home ext3 (rw,noatime)
```

**2. 运行前一个 Shell 命令，同时用 "bar" 替换掉命令行中的每一个 "foo"**

```
!!:gs/foo/bar
```

!! 表示重复执行上一条命令，并用 :gs/foo/bar 进行替换操作。

关于 !! 这个用法在前一篇文章中已有详细的介绍。

**3. 实时某个目录下查看最新改动过的文件**

```
watch -d -n 1 'df; ls -FlAt /path'
```

在使用这条命令时你需要替换其中的 /path 部分，watch 是实时监控工具，-d 参数会高亮显示变化的区域，-n 1 参数表示刷新间隔为 1 秒。

df; ls -FlAt /path 运行了两条命令，df 是输出磁盘使用情况，ls -FlAt 则列出 /path 下面的所有文件。

ls -FlAt 的参数详解：

> -F 在文件后面加一个文件符号表示文件类型，共有 */=>@| 这几种类型，* 表示可执行文件，/ 表示目录，= 表示接口( sockets) ，> 表示门， @ 表示符号链接， | 表示管道。

-l 以列表方式显示
-A 显示 . 和 ..
-t 根据时间排序文件

**4. 通过 SSH 挂载远程主机上的文件夹**

sshfs name@server:/path/to/folder /path/to/mount/point

这条命令可以让你通过 SSH 加载远程主机上的文件系统为本地磁盘，前提是你需要安装 FUSE 及 sshfs 这两个软件。

*译者注*：关于 sshfs 实际上我之前写过一篇文章介绍过，详见"在 Ubuntu 上使用 sshfs 映射远程 ssh 文件系统为本地磁盘"。

卸载的话使用 fusermount 命令：

fusermount -u /path/to/mount/point

**5. 通过 DNS 来读取 Wikipedia 的词条**

dig +short txt <keyword>.wp.dg.cx

这也许是最有趣的一条技巧了，David Leadbeater 创建了一个 DNS 服务器，通过它当你查询一个 TXT 记录类型时，会返回一条来自于 Wikipedia 的简短的词条文字，这是他的介绍。

这里有一个样例，来查询 "hacker" 的含义：

$ dig +short txt hacker.wp.dg.cx
"Hacker may refer to: Hacker (computer security), someone involved in computer security/insecurity, Hacker (programmer subculture), a programmer subculture originating in the US academia in the 1960s, which is nowadays mainly notable for the free software/" "open source movement, Hacker (hobbyist), an enthusiastic home computer hobbyist http://a.vu/w:Hacker"

这里使用了 dig 命令，这是标准的用来查询 DNS 的系统管理工具，+short 参数是让其仅仅返回文字响应，txt 则是指定查询 TXT 记录类型。

更简单的做法是你可以为这个技巧创建一个别名：

```
wiki() { dig +short txt $1.wp.dg.cx; }
```

然后试试吧：

```
$ wiki hacker
"Hacker may refer to: Hacker (computer security), ..."
```

如果你不想用 dig ，也可以用 host 命令：

```
host -t txt hacker.wp.dg.cx
```

**6. 用 Wget 的递归方式下载整个网站**

```
wget --random-wait -r -p -e robots=off -U Mozilla www.example.com
```

参数解释：

- -random-wait 等待 0.5 到 1.5 秒的时间来进行下一次请求
-r 开启递归检索
-e robots=off 忽略 robots.txt
-U Mozilla 设置 User-Agent 头为 Mozilla

其它一些有用的参数：

- -limit-rate=20K 限制下载速度为 20K
-o logfile.txt 记录下载日志
-l 0 删除深度（默认为5）
--wait=1h 每下载一个文件后等待1小时

**7. 复制最后使用的命令中的参数**

```
ALT + . (or ESC + .)
```

这个快捷键只能工作于 shell 的 emacs 编辑模式，它可以从最后使用的命令行中复制参数到当前命令行中，下面是一个样例：

```
$ echo a b c
a b c
$ echo <Press ALT + .>
$ echo c
```

你可以重复执行该快捷键，以便获取自己需要的参数，

以下是样例：

```
$ echo 1 2 3
1 2 3
$ echo a b c
a b c
$ echo <Press ALT + .>
$ echo c
$ echo <Press ALT + .> again
$ echo 3
```

另外，假如你想指定第1个或第2个，或者是第 n 个参数的话，可以按 ALT + 1 (或 ESC + 1) 或 ALT + 2 (或 ESC +2) 这样形式的快捷键。

以下是样例：

```
$ echo a b c
a b c
$ echo <Press ALT + 1> <Press ALT + .>
$ echo a
a
$ echo <Press ALT + 2> <Press ALT + .>
$ echo b
b
```

查看" Emacs Editing Mode Keyboard Shortcuts " 一文获取更多类似的快捷键。

**8. 执行一条命令但不保存到 history 中**

```
<space> command
```

这条命令可运行于最新的 Bash shell 里，在其它 shell 中没测试过。

通过在命令行前面添加一个空格，就可以阻止这条命令被保存到 bash history (~/.bash_history) 文件中，这个行为可以通过 $HISTIGNORE shell 变量来控制。我的设置是 HISTIGNORE="&:[ ]*" ，表示不保存重复的命令到 history 中，并且不保存以空格开头的命令行。$HISTIGNORE 中的值以冒号分隔。

如果你对此感兴趣，想深入了解的话，可进一步看此文"The Definitive Guide to Bash Command Line History"

**9. 显示当前目录中所有子目录的大小**

```
du -h --max-depth=1
```

- -max-depth=1 参数可以让 du 命令显示当前目录下 1 级子目录的统计信息，当然你也可以把 1 改为 2 ，进一步显示 2 级子目录的统计信息，可以灵活运用。而 -h 参数则是以 Mb 、G 这样的单位来显示大小。

*译者注*：在此推荐一个小工具 ncdu ，可以更方便的达到此效果。

**10. 显示消耗内存最多的 10 个运行中的进程，以内存使用量排序**

```
ps aux | sort -nk +4 | tail
```

显然这并不是最好的方法，但它确实用起还不错。

这是一个典型的管道应用，通过 ps aux 来输出到 sort 命令，并用 sort 排序列出 4 栏，再进一步转到 tail 命令，最终输出 10 行显示使用内存最多的进程情况。

假如想要发现哪个进程使用了大量内存的话，我通常会使用 htop 或 top 而非 ps 。

**额外的：用 python 快速开启一个 SMTP 服务**

```
python -m smtpd -n -c DebuggingServer localhost:1025
```

这是一个用 Python 标准库 smtpd （用 -m smtpd 指定) 实现在简易 SMTP 服务，运行于 1025 端口 。

另外三个参数的解释：

-n 参数让 Python 不要进行 setuid（改变用户）为 "nobody" ，也就是说直接用你的帐号来运行
-c DebuggingServer 参数是让 Python 运行时在屏幕上输出调试及运行信息
localhost:1025 参数则是让 Python 在本地的 1025 端口上开启 SMTP 服务

另外，假如你想让程序运行于标准的 25 的端口上的话，你必须使用 sudo 命令，因为只有 root 才能在 1-1024 端口上开启服务。如下：

```
sudo python -m smtpd -n -c DebuggingServer localhost:25
```

# === 第四部分 ===

**1.查看ASCII码表**

```
man 7 ascii
```

很多人初学编程都会接触到ascii码的概念，有时候为了查某个符号的ascii值，可能还得翻箱倒柜找出当年的课本？Linux Manpage里面其实包含了很多类似的实用资料，上述命令就能很详细的方式解释ascii编码，当然这里还有在线版。

man命令的第二个参数是区域码，用来区分索引词的范围，比如printf，在C标准库里面的printf跟bash当中的printf是不同的，前者的查询是man 3 printf，后者是man 1 printf。如果这个区域码省略，就会从1开始搜索，直到找到为止。

命令man man可以看到详细的解释。

manpages里面还有一些有趣而且实用的资料，可能鲜为人知：
man 1 intro - 一篇对从未接触过Linux的用户的简明教程。
man 2 syscalls - 内核系统请求的列表，按内核版本注释分类，系统编程必备。
man 2 select_tut - 关于select()系统请求的教程。
man 3 string - 在头文件内的所有函数。
man 3 stdio - 关于头文件的使用，标准输入/输出库的说明。
man 3 errno - 所有errorno的取值及说明。（C语言内类似其他语言的异常告知机制）
man 4 console_codes - Linux的终端控制码及其使用解释。

man 4 full - 介绍/dev/full这个总是处于"满"状态的磁盘。（对应/dev/null这个总是空的设备）
man 5 proc - 介绍/proc下的文件系统。
man 5 filesystems - 各种Linux文件系统。
第7区里面的资料通常最酷：
man 7 bootparam - 详细解释内核启动参数。
man 7 charsets - 解释各种语言的编码集。（gbk，gb2312等）
man 7 glob - 解释glob文件名管理机制的工作过程。
man 7 hier - 解释Linux文件系统结构各个部分的作用。
man 7 operator - C语言的运算符的列表。
man 7 regex - 介绍正则表达式。
man 7 suffixes - 常见文件后缀名的列表跟解释。
man 7 time - Linux的时钟机制解释。
man 7 units - 数值单位及其数值的解释。
man 7 utf8 - 描述UTF-8编码。
man 7 url - 解释URL、URI、URN等的标准。

**2.简易计时器**

```
time read
```

运行命令开始算起，到结束时按一下Enter，就显示出整个过程的时间，精确到ms级别。
time是用来计算一个进程在运行到结束过程耗费多少时间的程序，它的输出通常有三项：

```
$ time ls /opt
...

real 0m0.008s
user 0m0.003s
sys 0m0.007s
```

real指整个程序对真实世界而言运行所需时间，user指程序在用户空间运行的时间，
sys指程序对系统调用锁占用时间。

read本来是一个读取用户输入的命令，常见用法是read LINE，用户输入并回车后，
键入的内容就被保存到$LINE变量内，但在键入回车前，这个命令是一直阻塞的。

可见time read这命令灵活地利用了操作系统的阻塞。用这个命令来测试一壶水多久煮滚应该是不错的。

**3.远程关掉一台WINDOWS机器**

```
net rpc shutdown -I IP_ADDRESS -U username%password
```

Windows平台上的net命令是比较强大的，因为其后台是一个RPC类的系统服务，大家应该看过win下用net use file:///C|/ip/ipc%24 *这样一个命令建立IPC空连接，入侵主机的事情。

Linux下的net命令是samba组件的程序，通常包含在smbclient内，可以跟windows主机的文件、打印机共享等服务进行通讯，但是也支持rpc命令。

上述命令就是在远程Windows主机上执行了shutdown命令。当然这不一定成功，关系到win主机上面的安全设置。net命令能够控制到win主机就是了。

**4.在一个子SHELL中运行一个命令**

```
(cd /tmp && ls)
```

当然这只是演示，要查看目录当然可以ls /tmp。

好处就是不会改变当前shell的目录，以及如果命令中设计环境变量，也不会对当前shell有任何修改。

在Shell编程中还有很多使用上引号来括住一个命令：`ls /tmp`，这也是子shell过程。可是上引号的方法无法嵌套，而使用小括号的方法可以，一个比较纠结的例子是：

```
echo $(echo -e file:///C|/x%24%28printf "%x" 65))
```

**5.利用中间管道嵌套使用SSH**

```
ssh -t host_A ssh host_B
```

如果目标机器host_B处于比较复杂的网络环境，本机无法直接访问，但另外一台host_A能够访问到host_B，而且也能被本机访问到，那上述命令就解决了方便登录

host_B的问题。

但理论上这个过程是可以无限嵌套的，比如：

```
ssh -t host1 ssh -t host2 ssh -t host3 ssh -t host4 ...
```

嗯那神马FBI CIA的，有本事来捉我吧 ~

**6.清空屏幕**

```
<CTRL+l>;
```

这个跟之前介绍的reset命令重置终端的作用有些类似，其实都只是发送一段控制序列，让终端的显示复位。

还可以这样运行：

```
tput clear
```

tput是专门用来控制终端的一个小工具，也挺强大的，详细信息运行man tput查看。

**7.我想知道一台服务器什么时候重启完**

```
ping -a IP
```

系统管理员最常做的事情是重启系统。但是服务器的重启过程往往得花上好几分钟，什么你的服务器4个scsi卡？16个硬盘？系统是Redhat？还完全安装所有组件？好吧，它重启的时间都够你吃顿饭了，所以我很想知道它什么时候回来。

ping命令有个audible ping参数，-a，当它终于ping通你的服务器时会让小喇叭叫起来。

**8.列出你最常用的10条命令**

```
history | awk '{a[$2]++}END{for(i in a){print a[i] " " i}}' | sort -rn | head
```

这行命令组合得很妙： history输出用户了命令历史；awk统计并输出列表；sort排序；head截出前10行。

**9.检查GMAIL新邮件**

```
curl -u you@gmail.com --silent "https://mail.google.com/mail/feed/atom" |
perl -ne \
'
print "Subject: $1 " if /<title>(.+?)<\/title>/ && $title++;
print "(from $1)\n" if /<email>(.+?)<\/email>/;
'
```

Gmail的一个特色是支持Atom feed输出邮件列表，所以总是见到很多Gmail邮件提醒器之类的，因为开发特简单，atom很方便。

这里只是利用了perl的正则来解析atom（sed/awk也能做到）。

**10.用TELNET看《星球大战》**

```
telnet towel.blinkenlights.nl
```

没什么好解释的，就是ASCII艺术之一。如果你有ipv6连接，还能看到彩色版的。牛吧？

后记

说Shell是一种编程语言，可能有些尴尬，虽然很多人每天都在用Shell，但从来没见它荣登TIOBE编程语言排行榜之类的，可以说毫无名分，因为很多用户没意识到它是一种语言，只当做这是一个能够很好完成任务的工具，基本得理所当然，就好像GUI程序的菜单、按钮一样。

掌握Shell，通常能够让任务在数秒钟内完成，这就让Shell跟C、Perl、Python这些语言区别开来，没人否认后者更能胜任更多的任务，但是他们是在不同的层面上去做，Shell依赖大量的系统组件黏合调用，而后者依赖各种库，各所擅长不同的应用领域，比喻就是，Shell是混凝土，可以很方便地粘合一些建筑组件而成为稳固的高楼大厦；但同样是粘合剂，粘玻璃窗、粘书报、粘皮鞋，混凝土是绝对不合适的，Shell并不擅长一些细致操作，比如它连浮点运算都不支持，更别提什么图形运算什么的。但这并不妨碍Shell来帮我们完成很多粗重任务。

Shell的工作方式，大多数入门用户会觉得枯燥难学，而所谓的经典教材也离不开《Advanced Bash-Scripting》、《Bash Guide for Beginners》，但类似本文这样的一些"雕虫小技"因为难登大雅之堂绝不会收录进去。这情况如果象国外一些unix用户比较多的地方会有很好改善，即使是新手，偶尔看看别人的操作都能"偷师"一手，我编译本系列文章其实也就希望稍微改善一下这个状况。